NTNU

TDT4136 - Introduction to Artificial Intelligence

Exercise 3

Mathias Ose & Øyvind Robertsen

October 3, 2014

1 | Using the A* Algorithm

1.1 Problem A: Pathfinding in 2D Games

All deliverables in the form of code are enclosed.

Subproblem A.1: Grids with Obstacles

The following figures show the solution path calculated by the A* implementation, with the OPEN and CLOSED sets visualized as respectively cyan and pink dots.

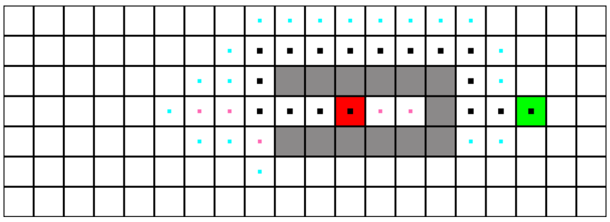

Figure 1.1: Board 1.1

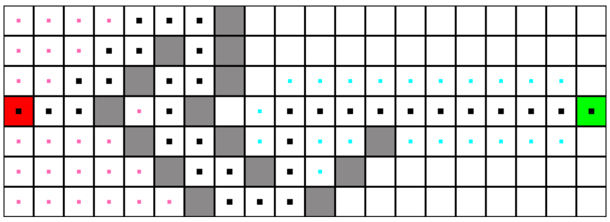

Figure 1.2: Board 1.2

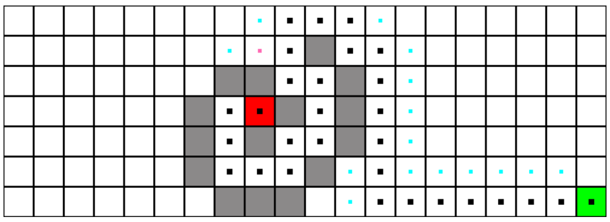

Figure 1.3: Board 1.3

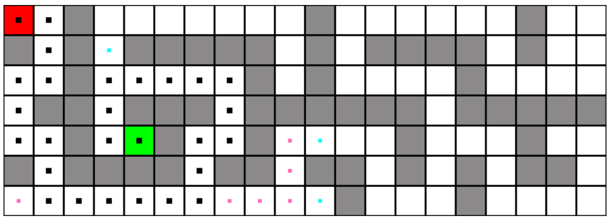
Figure 1.4: Board 1.4

Subproblem A.2: Grids with different cell costs

For this subproblem, we modified our implementation to be able to parse weighted boards as well as correctly handling cost calculation of weighted nodes. Again, the visualizations include the OPEN and CLOSED sets.

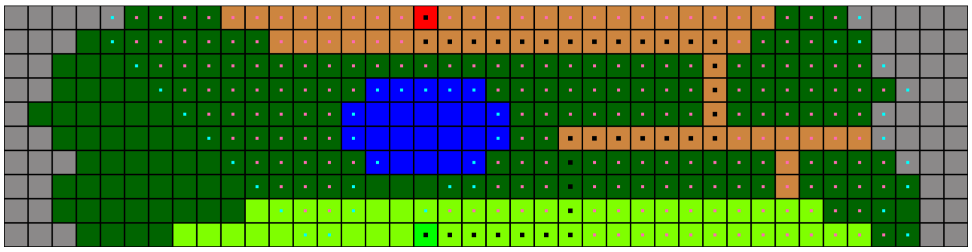
Figure 1.5: Board 2.1

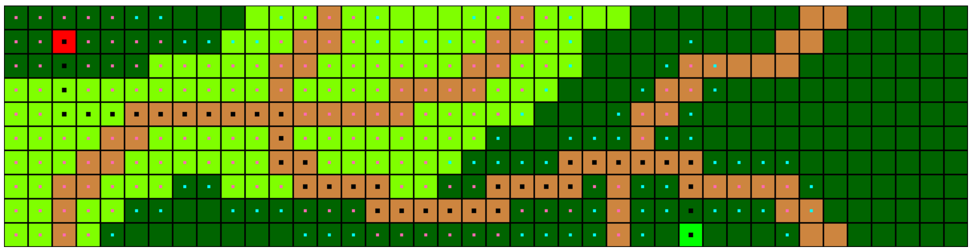
Figure 1.6: Board 2.2

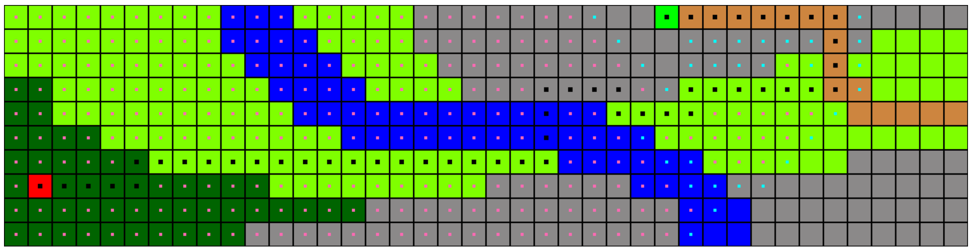
Figure 1.7: Board 2.3

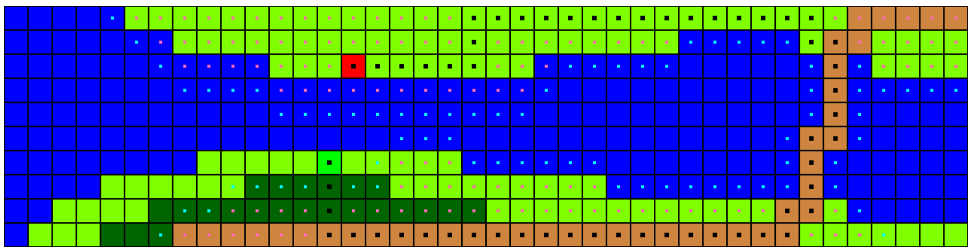

Figure 1.8: Board 2.4

Comparison with BFS and Dijkstra's Algorithm

Board 1.1

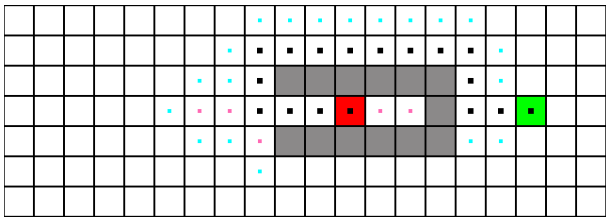

Figure 1.9: Board 1.1 - A*

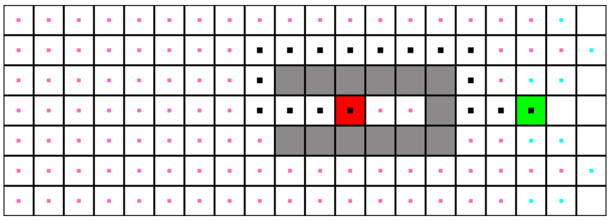

Figure 1.10: Board 1.1 - Dijkstra

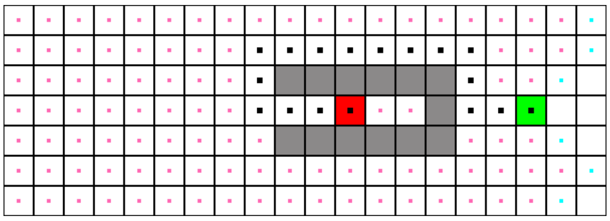

Figure 1.11: Board 1.1 - BFS

All three algorithms find an equally short path, but A* examines significantly fewer nodes doing it.

Board 1.2

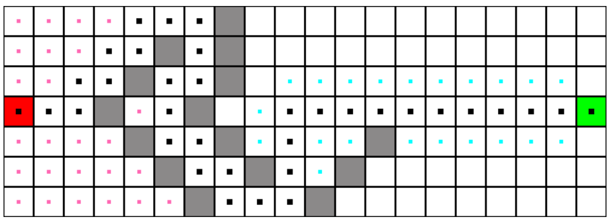
Figure 1.12: Board 1.2 - A*

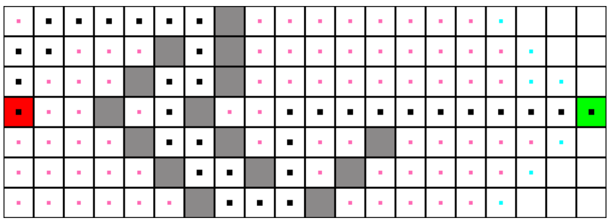
Figure 1.13: Board 1.2 - Dijkstra

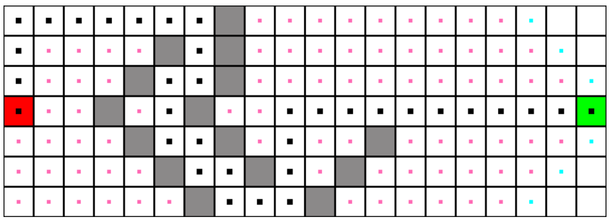
Figure 1.14: Board 1.2 - BFS

All algorithms find the shortest path, but again, A* is much more efficient.

Board 1.3

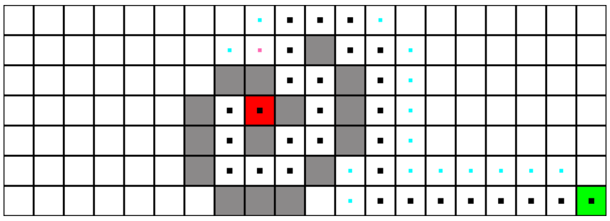
Figure 1.15: Board 1.3 - A*

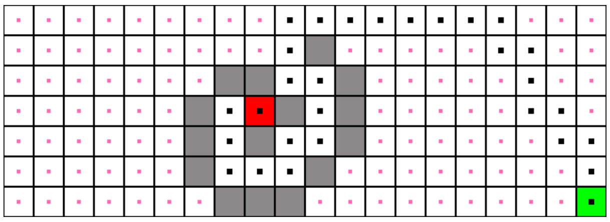
Figure 1.16: Board 1.3 - Dijkstra

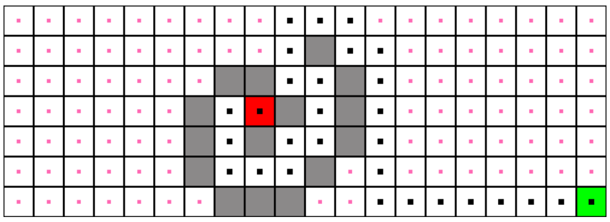
Figure 1.17: Board 1.3 - BFS

Again, A* is much more efficient than the others.

Board 1.4

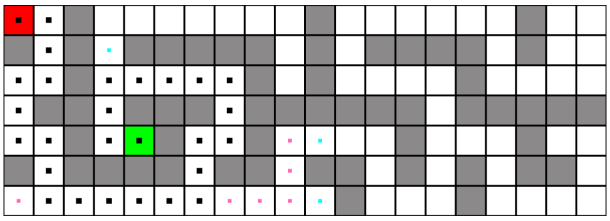
Figure 1.18: Board 1.4 - A*

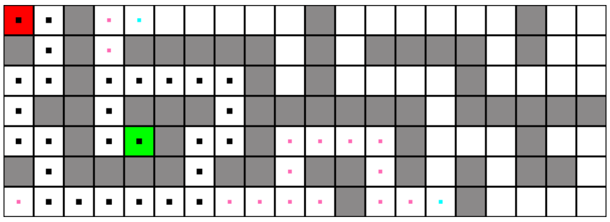
Figure 1.19: Board 1.4 - Dijkstra

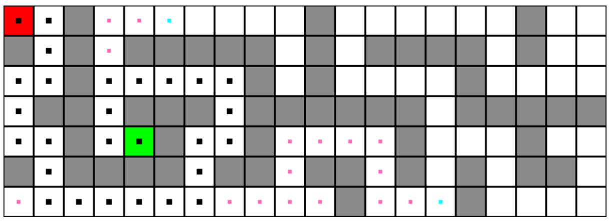
Figure 1.20: Board 1.4 - BFS

For this board, we see that there’s not much difference between the three algorithms. A* explores marginally fewer nodes than the other two.

Board 2.1

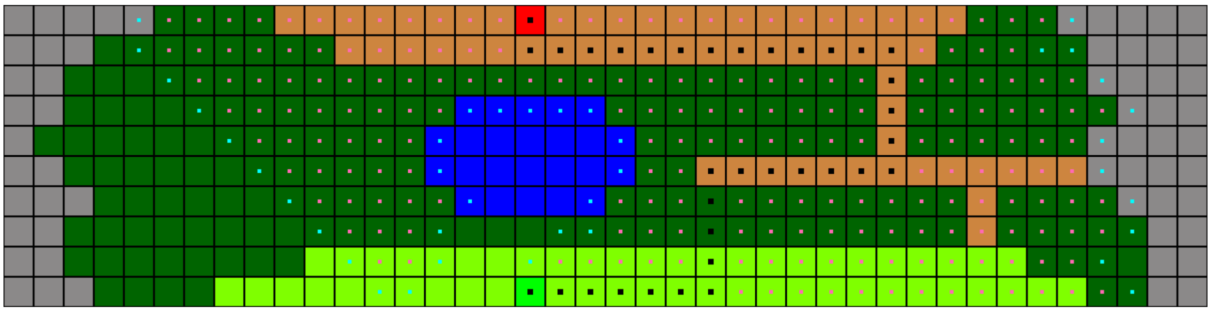
Figure 1.21: Board 2.1 - A*

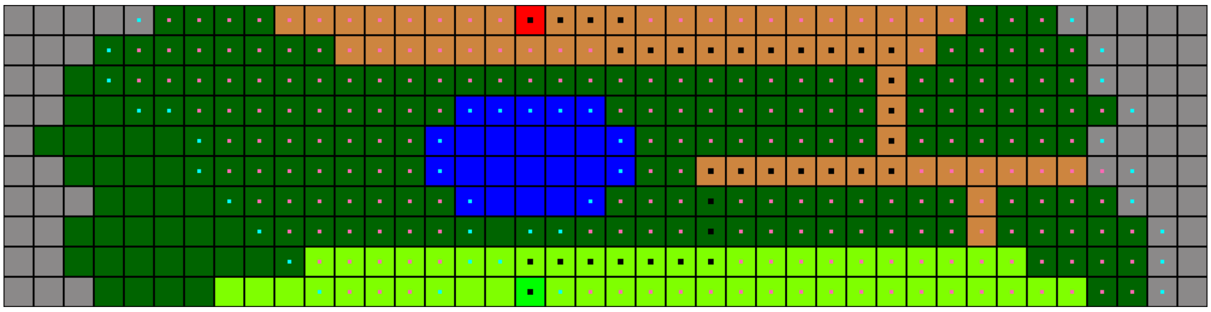
Figure 1.22: Board 2.1 - Dijkstra

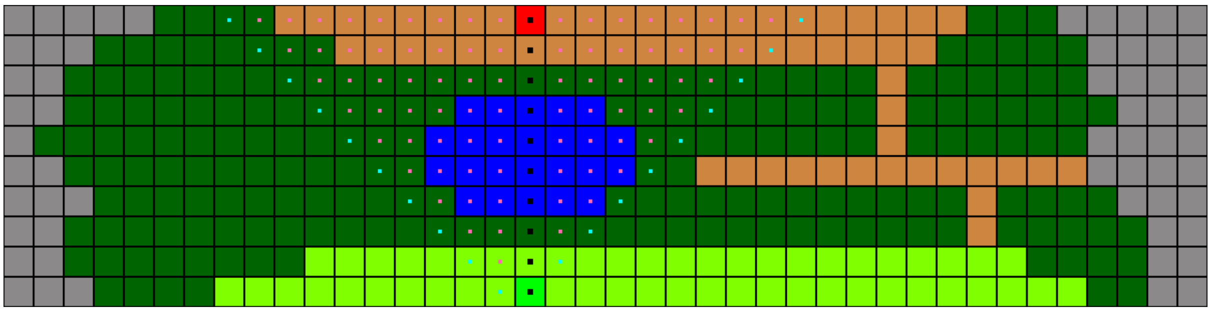
Figure 1.23: Board 2.1 - BFS

For this board, we see that A* and Djikstra both find equally optimal paths, both processing about the same number of nodes. (A* slightly lower as the heuristic is taken into consideration.) BFS processes fewer nodes, but finds an extremely unoptimized path.

Board 2.2

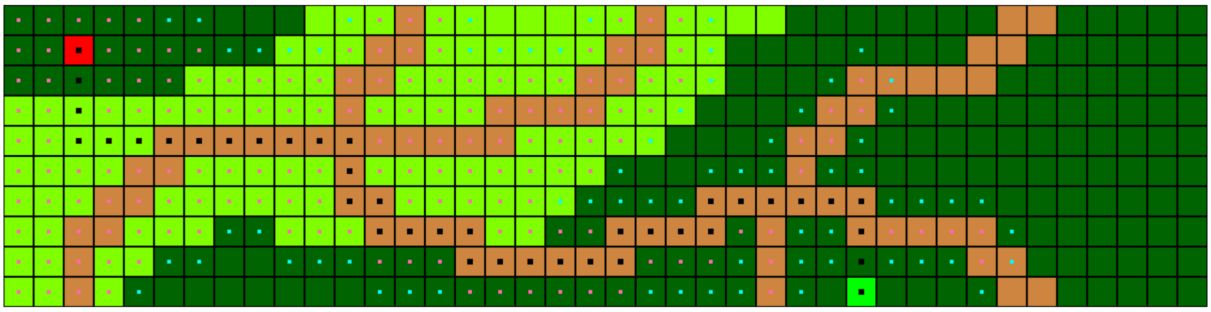
Figure 1.24: Board 2.2 - A*

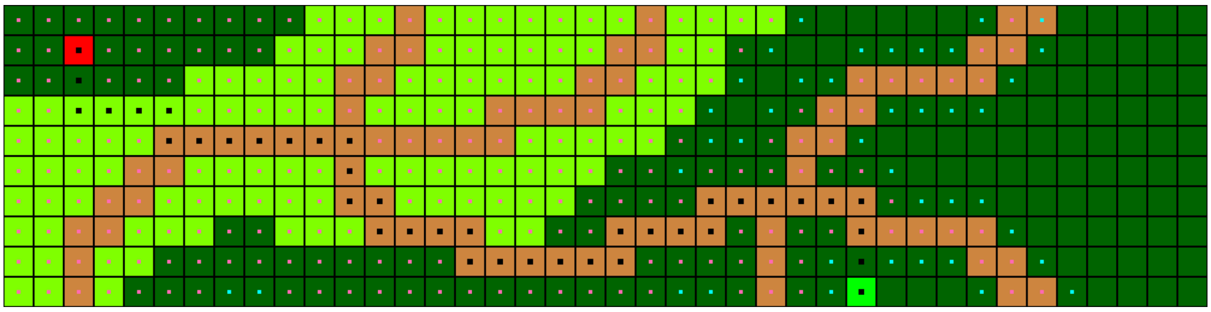
Figure 1.25: Board 2.2 - Dijkstra

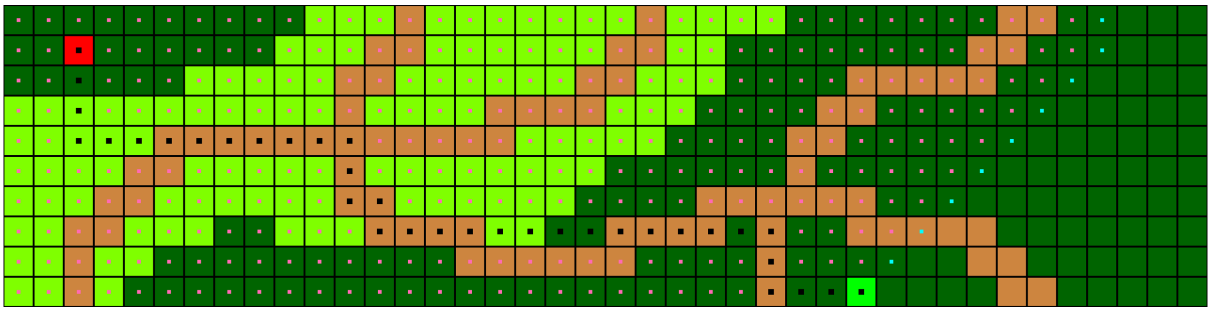
Figure 1.26: Board 2.2 - BFS

A* and Dijkstra both perform reasonably well here, while BFS makes quite a few suboptimal choices while also processing more nodes.

For the case of A* vs. Djikstra, we see that A* (as it should) refrains from processing nodes that have low cell weight, but also have higher heuristic values.

Board 2.3

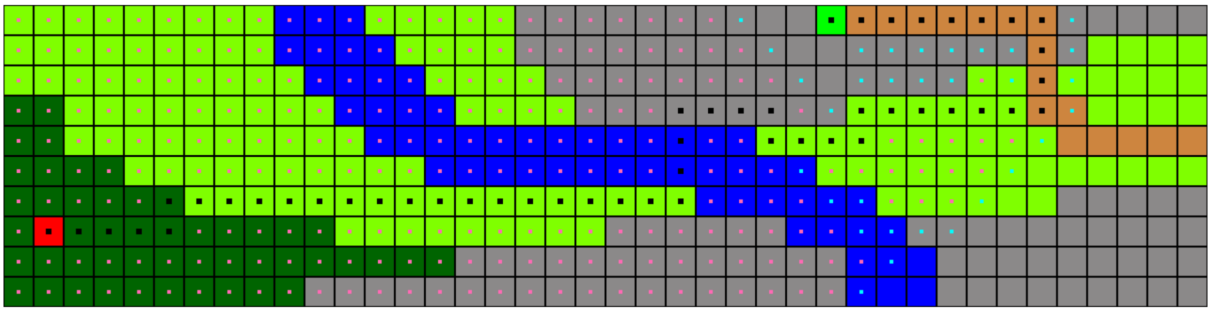

Figure 1.27: Board 2.3 - A*

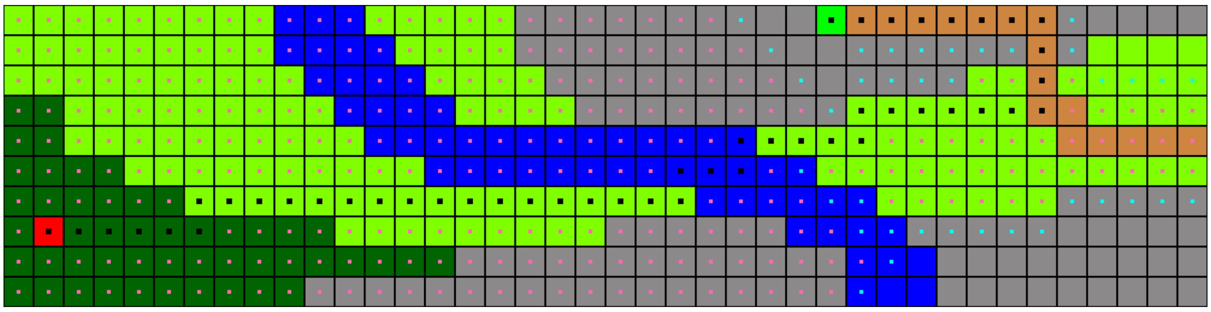

Figure 1.28: Board 2.3 - Dijkstra

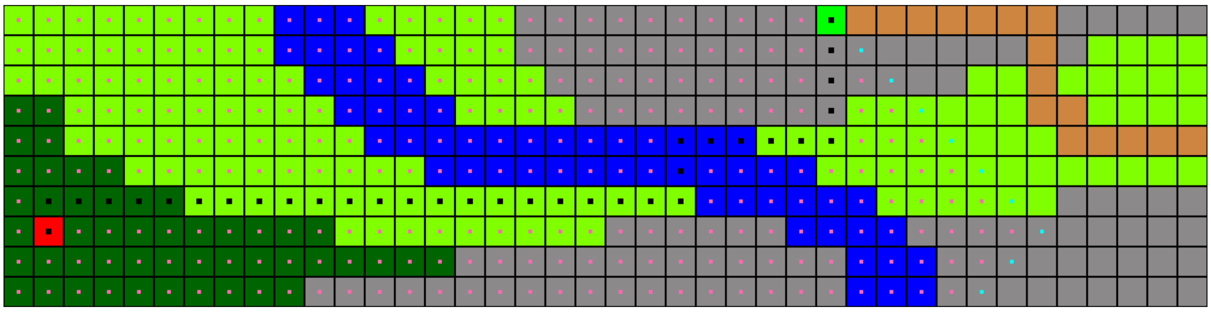

Figure 1.29: Board 2.3 - BFS

Again, A* and Dijkstra both find equally optimal paths, while BFS naively picks the first path it can find.

Board 2.4

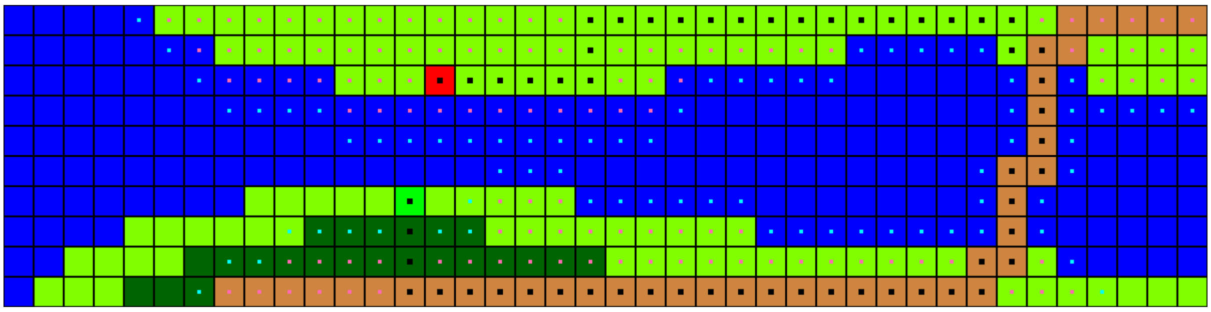

Figure 1.30: Board 2.4 - A*

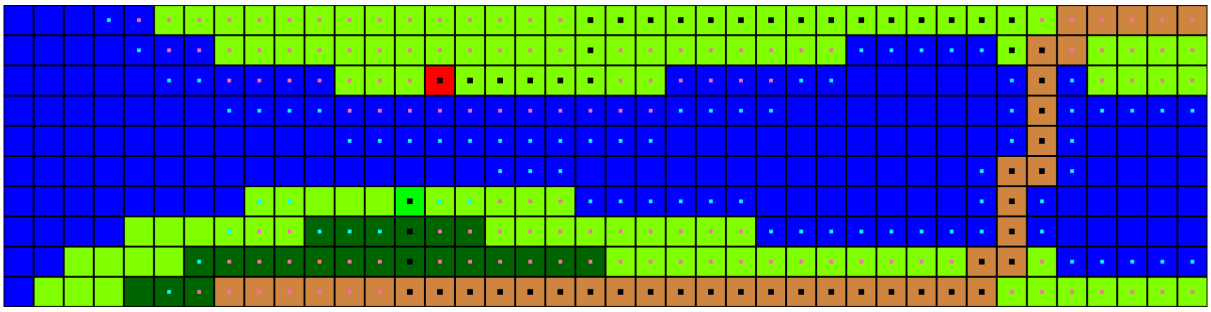

Figure 1.31: Board 2.4 - Dijkstra

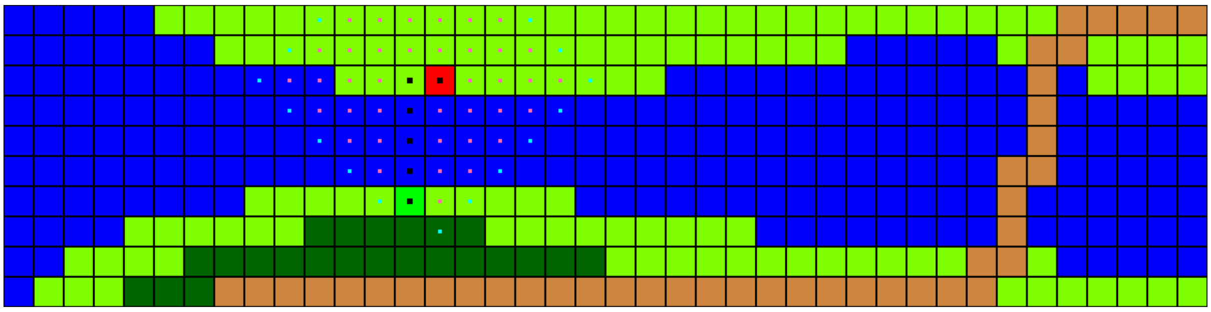

Figure 1.32: Board 2.4 - BFS

For this final board, A* and Dijkstra find the same path. A* processes slightly fewer nodes, notably those in the bottom right corner of the board. BFS processes few nodes, but finds a suboptimal path.